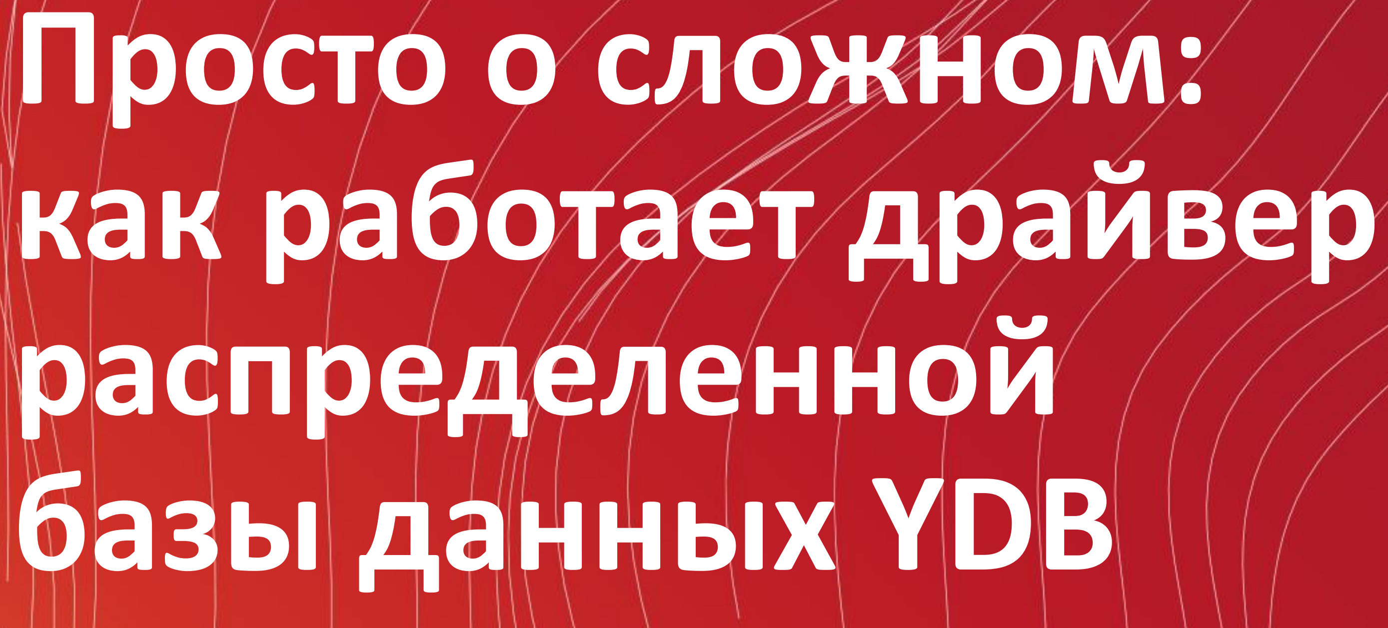


# Просто о сложном: как работает драйвер распределенной базы данных YDB

Мясников Алексей,
старший разработчик в команде YDB

HL++ HighLoad++ 2022

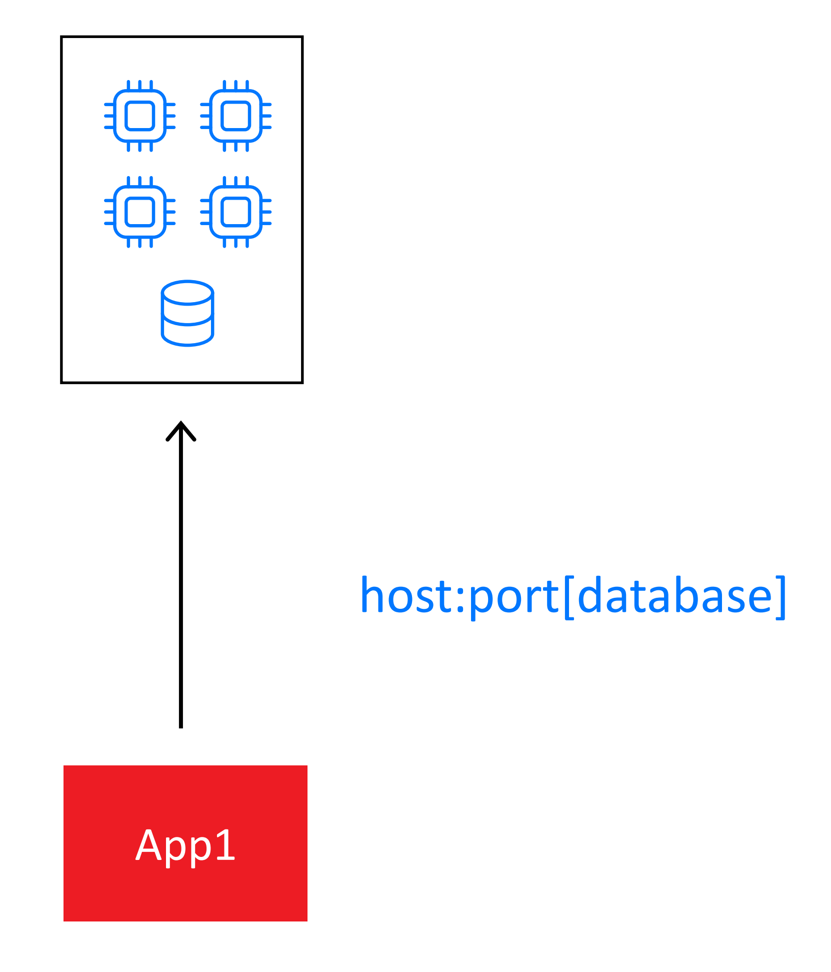
Яндекс

# Содержание

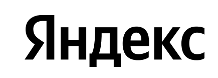
01

# Традиционные базы данных

# Маленькая база данных








# Растет нагрузка

host:port[database]

App1 App2 App3 App4 App5 App6 App7 App8





# Выходы из ситуации

host:port[database]

App1 App2 App3 App4 App5 App6 App7 App8

# Хосты БД в конфиге

host1:port1[database]
host2:port2[database]
host3:port3[database]

App1 App2 App3 App4 App5 App6 App7 App8




# Хосты БД в конфиге: добавляем хост

host1:port1[database]
host2:port2[database]
host3:port3[database]
host3:port3[database]

App1 App2 App3 App4 App5 App6 App7 App8





# Хосты БД в конфиге: удаляем хост

host1:port1[database]
~~host2:port2[database]~~
host3:port3[database]
host3:port3[database]

App1 App2 App3 App4 App5 App6 App7 App8





# Хосты БД за DNS

host:port[database]

host = [IP1,IP2,IP3]

App1 App2 App3 App4 App5 App6 App7 App8





# DNS: добавляем хост

host:port[database]

host = [IP1,IP2,IP3, IP4]

App1 App2 App3 App4 App5 App6 App7 App8

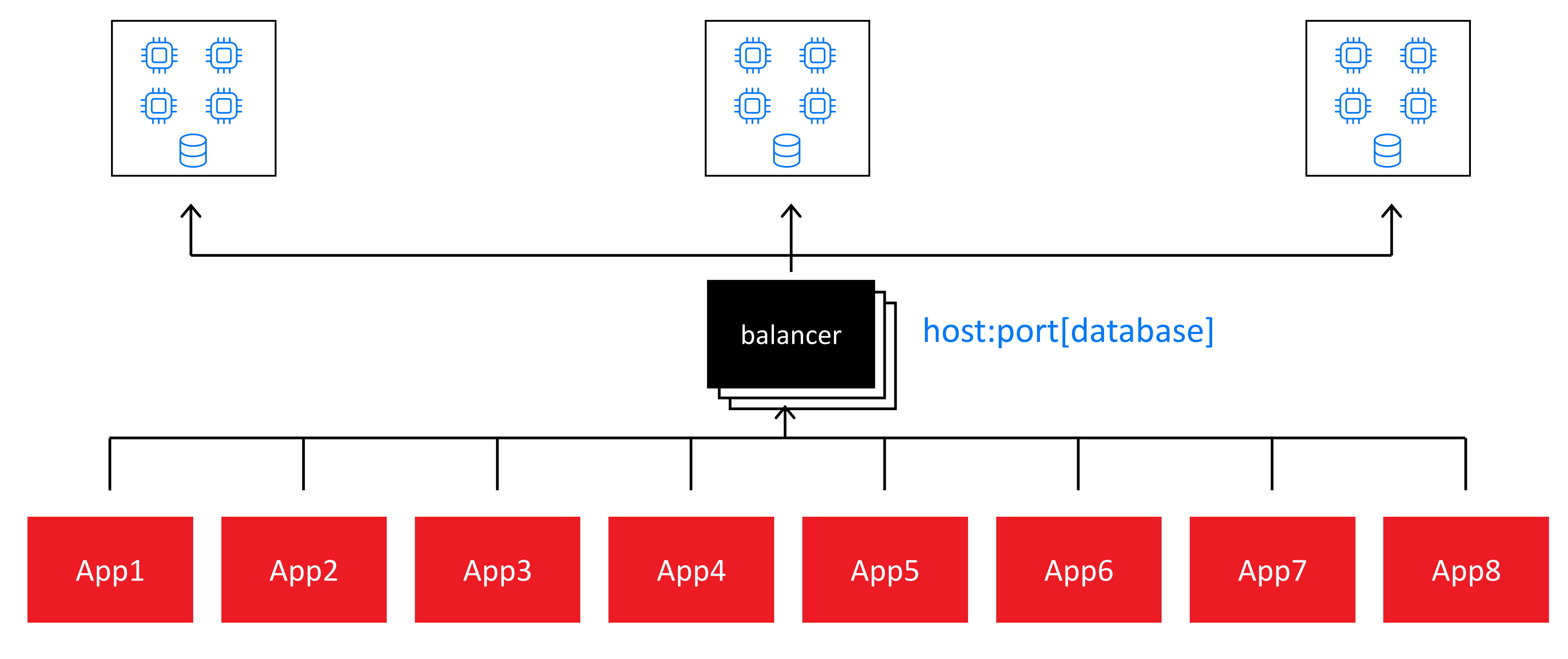
# DNS: удаляем хост

host:port[database]

host = [IP1,~~IP2~~,IP3, IP4]

App1 App2 App3 App4 App5 App6 App7 App8

# Хосты БД за балансером (L3, L7)



# Хосты БД за балансером (L3, L7): добавляем хост

balancer

host:port[database]

App1 App2 App3 App4 App5 App6 App7 App8

# Хосты БД за балансером (L3, L7): удаляем хост

balancer

host:port[database]

App1 App2 App3 App4 App5 App6 App7 App8





02

# YDB way

# YDB

Распределённая open-source SQL-база данных для операционных нагрузок

+ SQL для OLTP
+ Горизонтальное масштабирование
+ Транзакции с гарантиями ACID в нескольких AZ
+ Работоспособность и автоматическое восстановление при отказах
+ Масштабирование на миллионы транзакций в секунду и сотни терабайт данных
+ Поддерживаются OLAP-сценарии,
  координация распределённых систем (like ZooKeeper),
  доставка сообщений (like Kafka)

https://ydb.tech

https://github.com/ydb-platform/ydb

https://t.me/ydb_ru

https://t.me/ydb_en

# Кластер YDB с точки зрения клиента

AZ1 AZ2 AZ3

...

App1 App2 App3 App4 App5 App6 App7 App8





# Что такое нода YDB?

***Исполнительный узел в кластере YDB***

+ виртуальная машина
+ pod kubernetes
+ отдельный самостоятельный процесс

***Особенности:***

1. однородность
2. волатильность
3. разные для разных баз данных
4. fqdn + порт
5. grpc API

# Динамическое расширение

AZ1

AZ2

AZ3

...

App1

App2

App3

App4

App5

App6

App7

App8





# Освобождение лишних ресурсов

AZ1

AZ2

AZ3

...

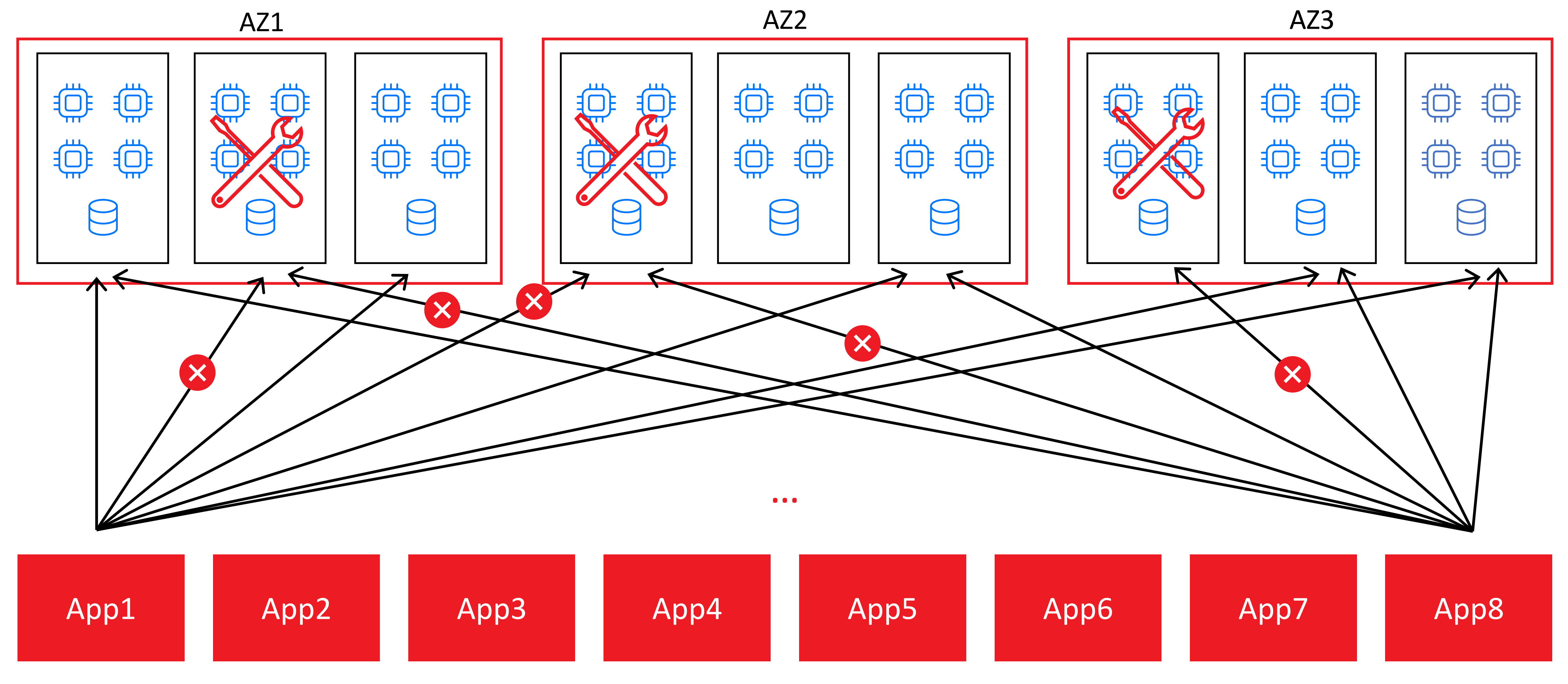
App1 App2 App3 App4 App5 App6 App7 App8

# Отказы в большой системе - норма



+ ***Диски***
  - Время наработки на отказ (MTTF) одного диска — 1.4 млн часов или 160 лет
  - Время наработки на отказ >= 1 диска из ~1000 серверов по 4 диска в каждом — 180 часов
    или 8 дней
+ ***Серверы***
  - Отказы железа
  - Отключение серверов/стоек для обслуживания
  - Выход из строя целых стоек (ToR switch, питание)
  - Отключение дата-цетров (учения, питание, сеть)

# Обслуживание при режиме эксплуатации 24/7



# 03

# Требования к драйверу

# Отказ ноды

AZ1

AZ2

AZ3

...

App1 App2 App3 App4 App5 App6 App7 App8

# Отказ ДЦ

AZ1

AZ2

AZ3

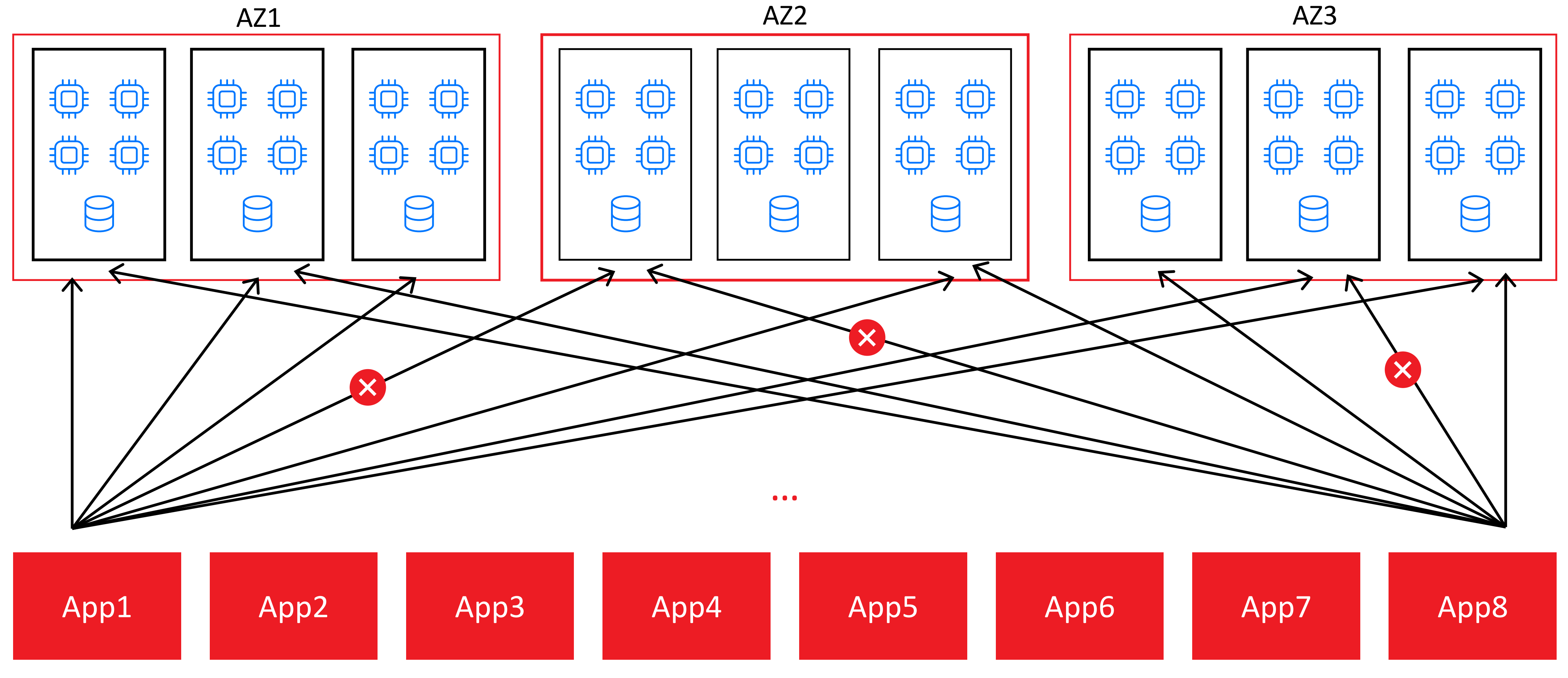
...

App1 App2 App3 App4 App5 App6 App7 App8





# Сетевые ошибки



# Серверные ошибки

AZ1

AZ2

AZ3

...

App1 App2 App3 App4 App5 App6 App7 App8

# Клиентская и серверная балансировка запросов

Низкая загрузка Умеренная загрузка Высокая загрузка

AZ1 AZ2 AZ3

...

App1 App2 App3 App4 App5 App6 App7 App8





# Пессимизация соединений

Errors RPS

2.50 reg/s

2 reg/s

1.50 reg/s

1 reg/s

0.500 reg/s

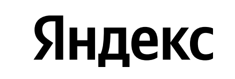
0 reg/s

19:25:40 19:26:50 19:27:40

2021-12-29 19:26:45

transport/unavailable [000014]: 2.44 req/s








# Пессимизация соединений

Connections usage

80
70
60
50
40
30
20
10
0

19:25:30
19:27:00

**2021-12-29 19:27:00**

| | |
|---|---|
| kikimr0425.search.yandex.net:2135 | 0 |
| kikimr0425.search.yandex.net:31051 | 2 |
| kikimr0447.search.yandex.net:31051 | 2 |
| kikimr0482.search.yandex.net:31051 | 23 |
| kikimr0518.search.yandex.net:31051 | 28 |
| kikimr0602.search.yandex.net:31051 | 25 |





# 04

## YDB API

# gRPC-сервисы YDB на ноде YDB

Table

Coordination

Discovery

RateLimiter

Monitoring

Scheme

Topic

Auth

ScriptingYQL

...





# YDB API

Table
ydb_table_v1.proto

Coordination
ydb_coordination_v1.proto

Discovery
ydb_discovery_v1.proto

RateLimiter
ydb_ratelimiter_v1.proto

Monitoring
ydb_monitoring_v1.proto

Scheme
ydb_scheme_v1.proto

Topic
ydb_topic_v1.proto

Auth
ydb_auth_v1.proto

...

ScriptingYQL
ydb_scripting_v1.proto



https://github.com/ydb-platform/ydb-api-protos/





# YDB API

Table

Coordination

Discovery

RateLimiter

Monitoring

Auth

Scheme

Topic

...

ScriptingYQL





# Сервис Discovery

Нода1

Нода2

Нода3

Нода4

...

AZ1

...

AZ2

...

AZ3

- БД1
- БД1
- БД1
- БД1



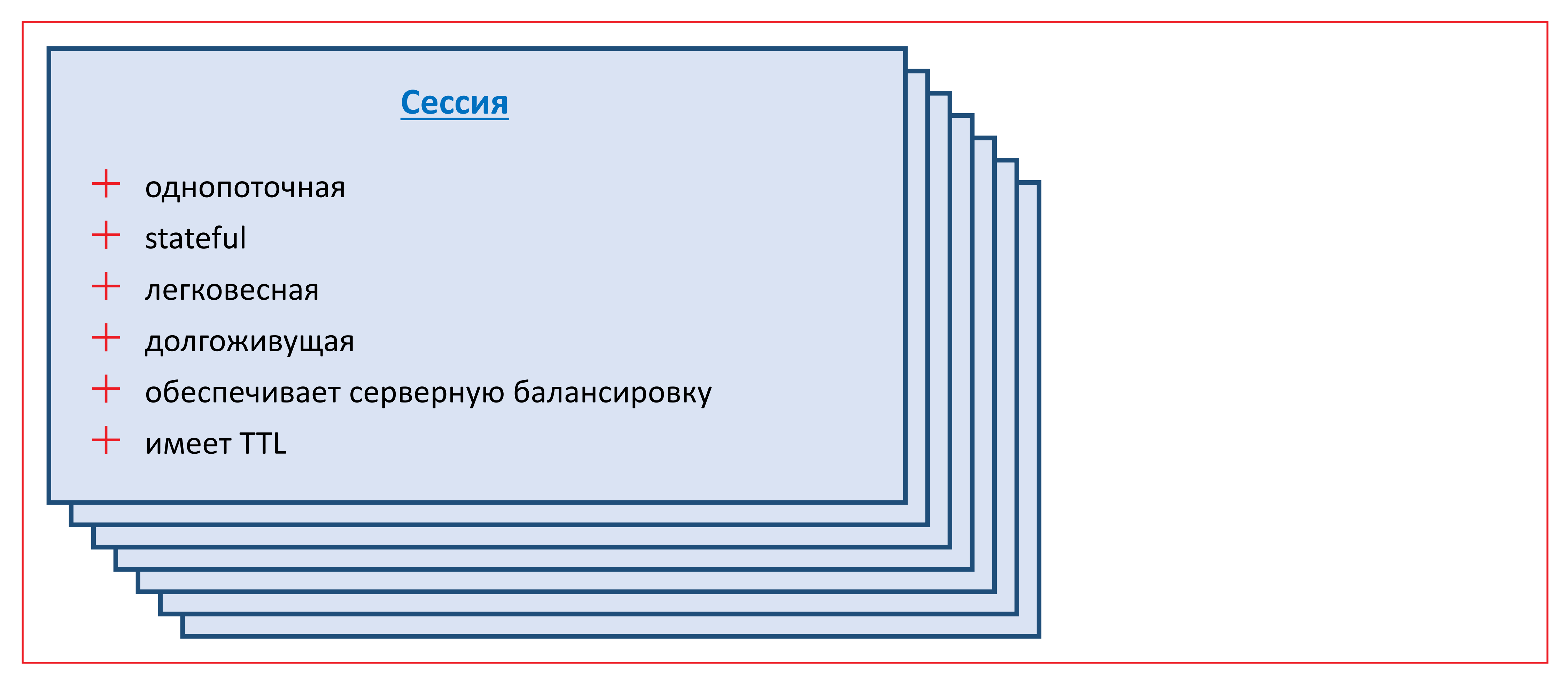


# Чтобы начать работать с YDB

1. Выяснить конфигурацию кластера через специальный запрос **Discovery/ListEndpoints**, указав имя базы данных
2. Подключиться напрямую к нодам YDB



# Сервис таблиц на ноде YDB



05

# Жизненный цикл драйвера YDB

# Исходные данные:

**1) начальный Endpoint**

grpcs://ydb.serverless.yandexcloud.net:2135

grpcs://lb.etnt8n4t.ydb.mdb.yandexcloud.net:2135

+ Прокси-сервис (балансер)
+ DNS-запись с IP-адресами нод YDB

**2) имя базы данных**

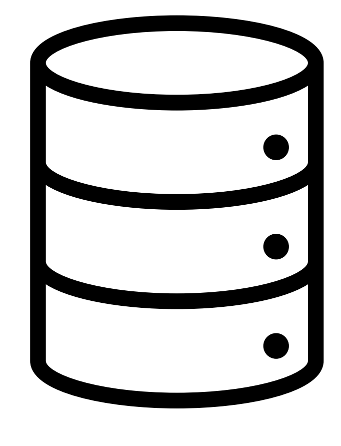
/ru-central1/b1g8spbalf3s/etntdt2dn44t

# 1. Инициализация драйвера

# 1.1 Выясняем конфигурацию кластера

Discovery/ListEndpoints(Database)

[Endpoint1, Endpoint2, …, EndpointN]

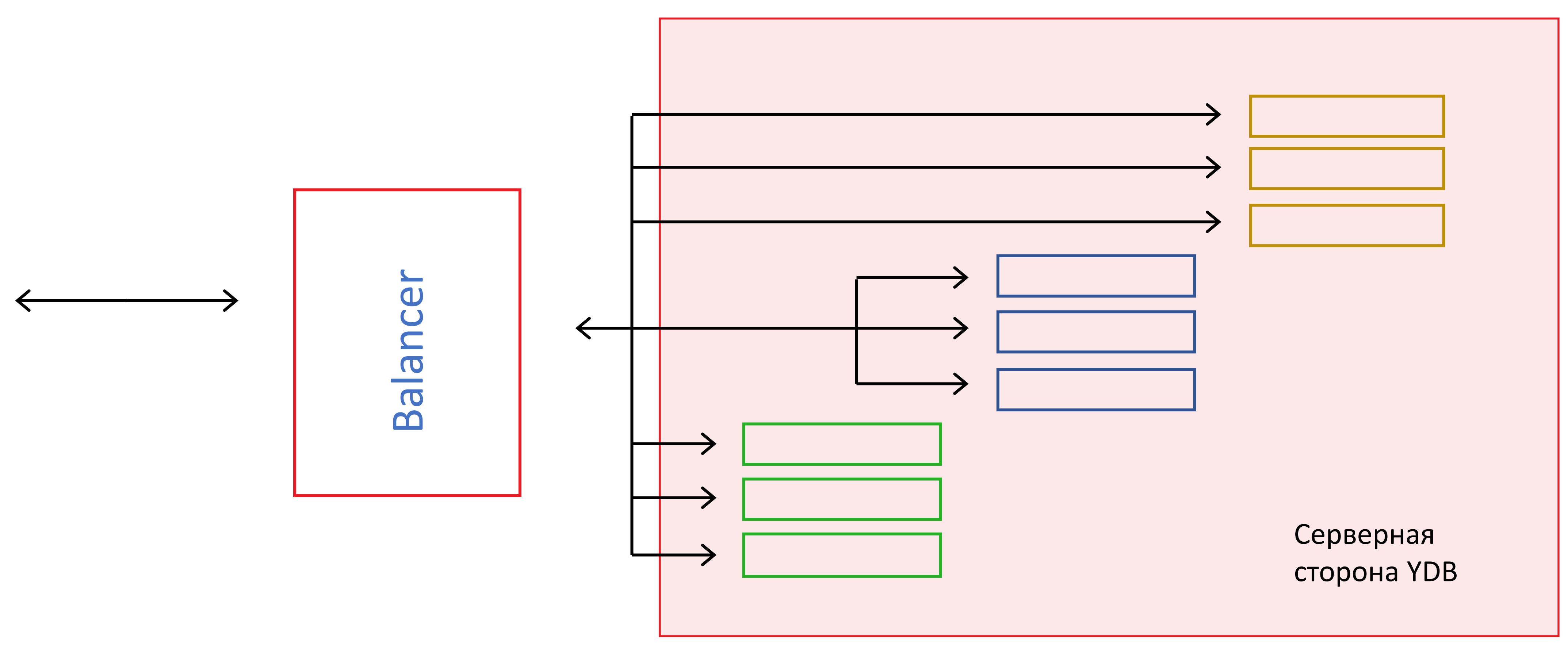


начальный Endpoint





# 1.2 Инициализируем клиентский балансировщик



# 1.3 Запускаем фоновый *Discovery* и применение результатов в балансировщике





# Драйвер инициализирован

# 2. Выполнение табличных запросов

```
SELECT `title`
FROM `/highload/2022/november/reports`
WHERE speaker=$speakerName;
```

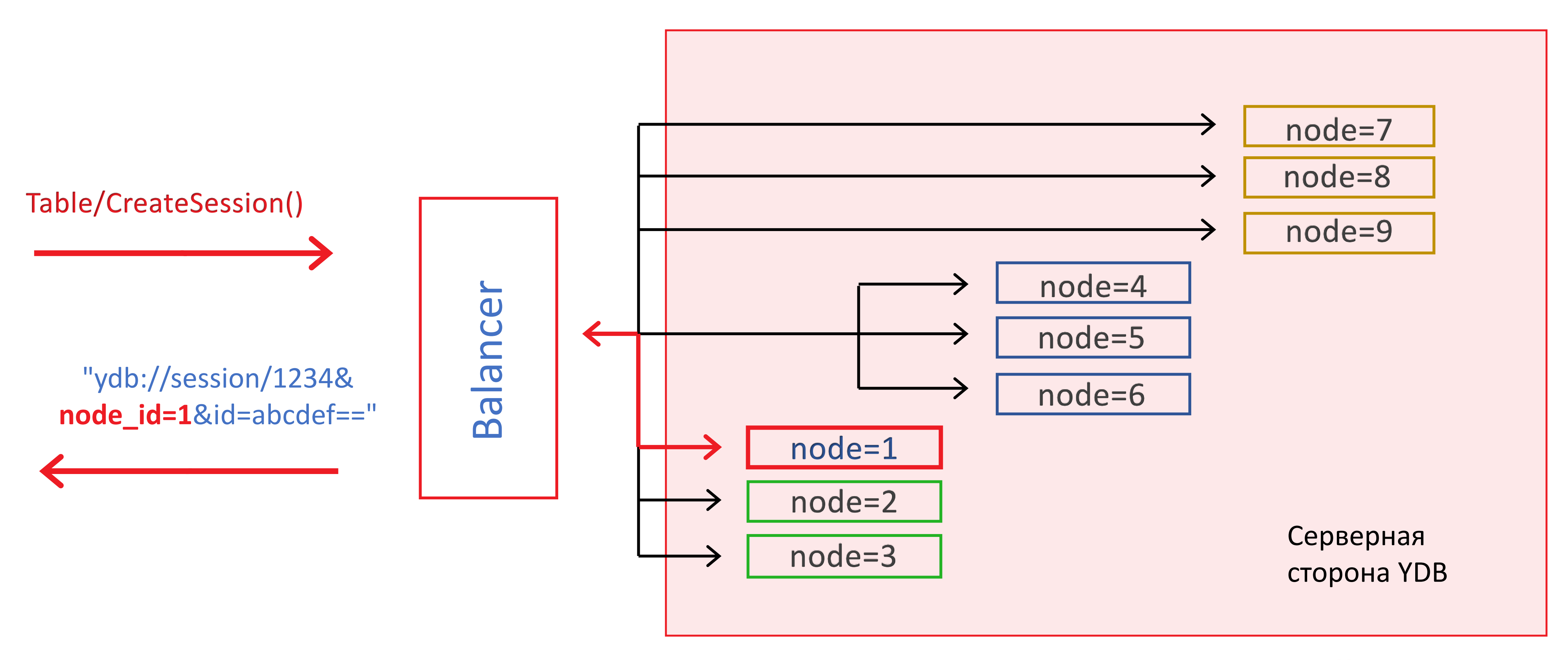
# Простейшая реализация:

1. Создать сессию

2. Выполнить запрос на сессии

3. Закрыть сессию

# 2.1 Создание сессии



# 2.2 Выполнение запроса

Table/ExecuteDataQuery(
 "ydb://session/1234&**node_id=1**&id=abcdef==",
 query,
 $speackerName="Мясников Алексей",
)

title="Просто о сложном:
как работает драйвер
распределенной
базы данных YDB"

Balancer

node=7
node=8
node=9
node=4
node=5
node=6
node=1
node=2
node=3

Серверная
сторона YDB





# 2.1 Сессия может быть создана на другой ноде

Table/CreateSession()

"ydb://session/5678&
**node_id=4**&id=abcdef=="

Balancer

node=7
node=8
node=9
node=4
node=5
node=6
node=1
node=2
node=3

Серверная
сторона YDB





# 2.2 Запросы на сессии следует отправлять на «правильную» ноду

```
Table/ExecuteDataQuery(
 "ydb://session/1234&node_id=4&id=abcdef==",
  query,
  $speackerName="Мясников Алексей",
)
```

title="Просто о сложном: как работает драйвер распределенной базы данных YDB"

Balancer

node=7
node=8
node=9
node=4
node=5
node=6
node=1
node=2
node=3

Серверная сторона YDB





# 2.2 Если запрос ушел на «неправильную» ноду

Table/ExecuteDataQuery(
"ydb://session/1234&**node_id=4**&id=abcdef==",
query,
$speackerName="Мясников Алексей",
)

title="Просто о сложном: как работает драйвер распределенной базы данных YDB"

Balancer

node=7
node=8
node=9
node=4
node=5
node=6
node=1
node=2
node=3

Серверная сторона YDB





# 2.3 Закрыть сессию

Table/CloseSession(
"ydb://session/1234&**node_id=1**&id=abcdef==",
)







# Что можно оптимизировать?

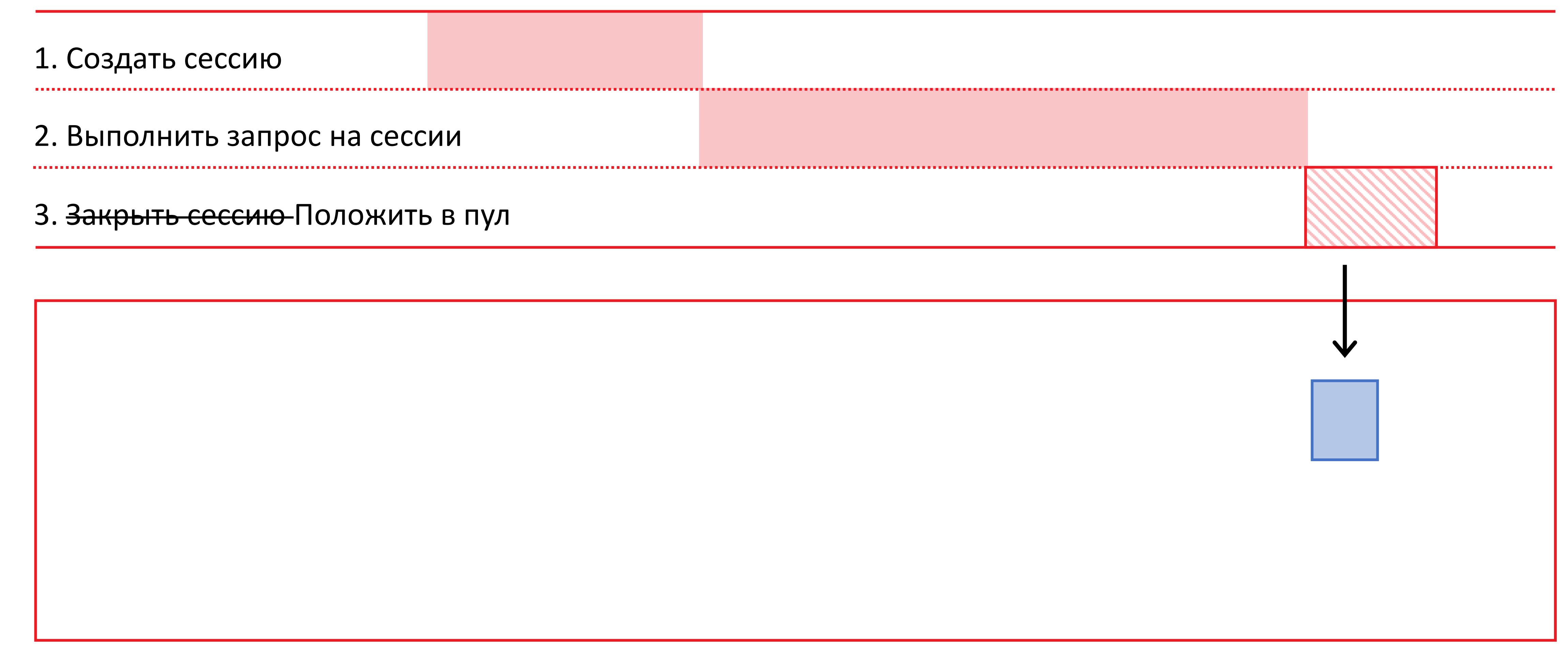
| | | | |
|---|---|---|---|
| 1. Создать сессию | ? | | |
| 2. Выполнить запрос на сессии | | | |
| 3. Закрыть сессию | | | ? |



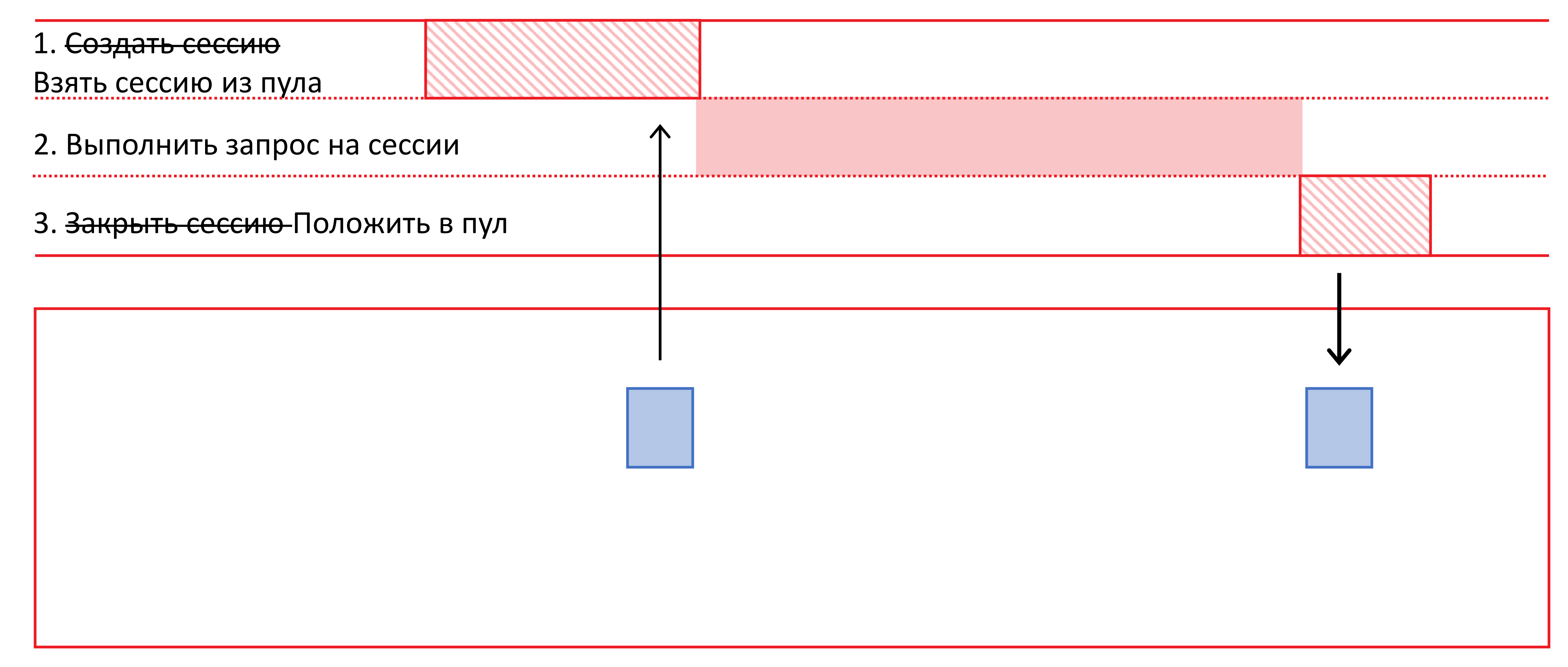


# 3. Пул сессий

# 3.1 Работа с пулом:



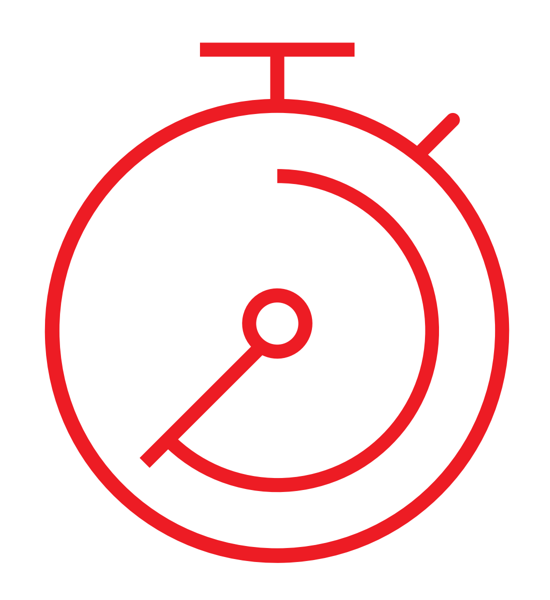
# 3.1 Работа с пулом:



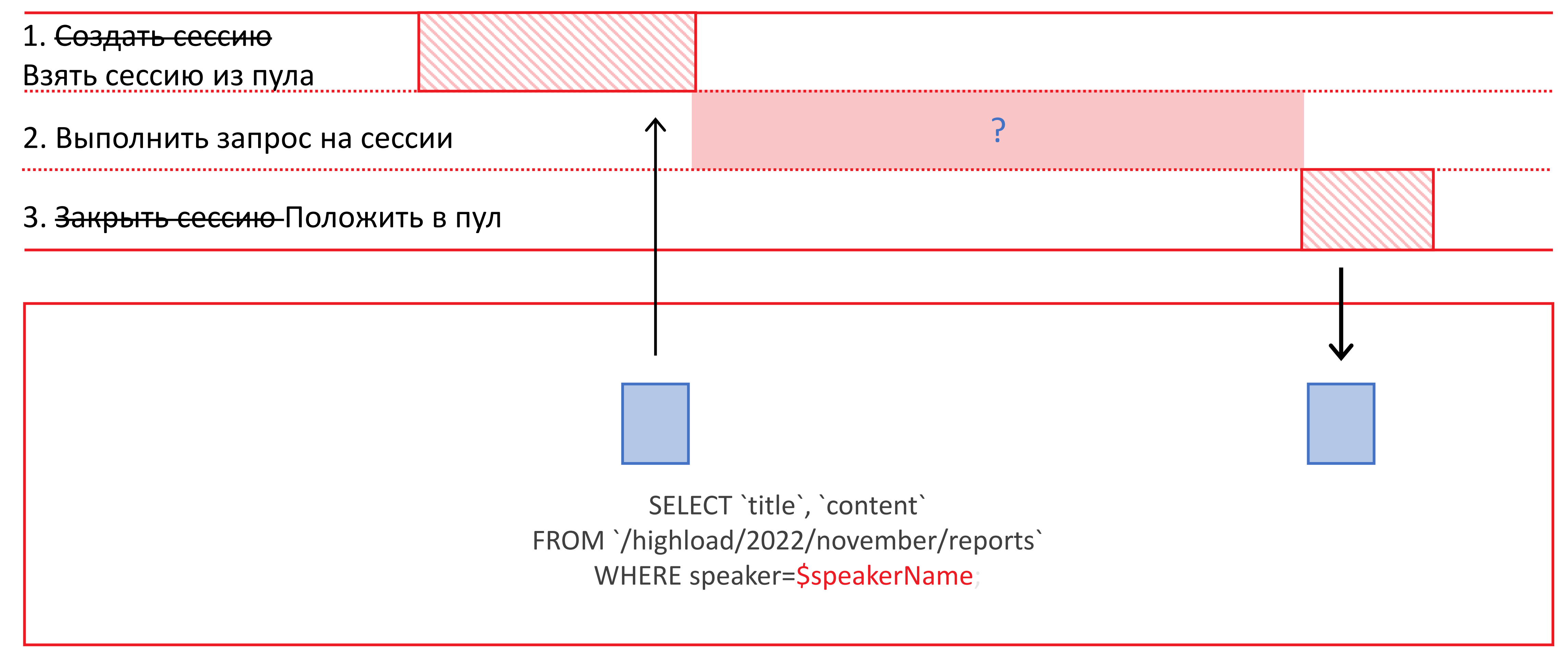
# Время жизни сессии

| | | | | |
|---|---|---|---|---|
| Сессия | Сессия | Сессия | | Сессия |
| Сессия | Сессия | Сессия | … | Сессия |
| Сессия | Сессия | Сессия | … | Сессия |
| Сессия | Сессия | Сессия | … | Сессия |
| … | … | … | | … |
| Сессия | Сессия | Сессия | … | Сессия |

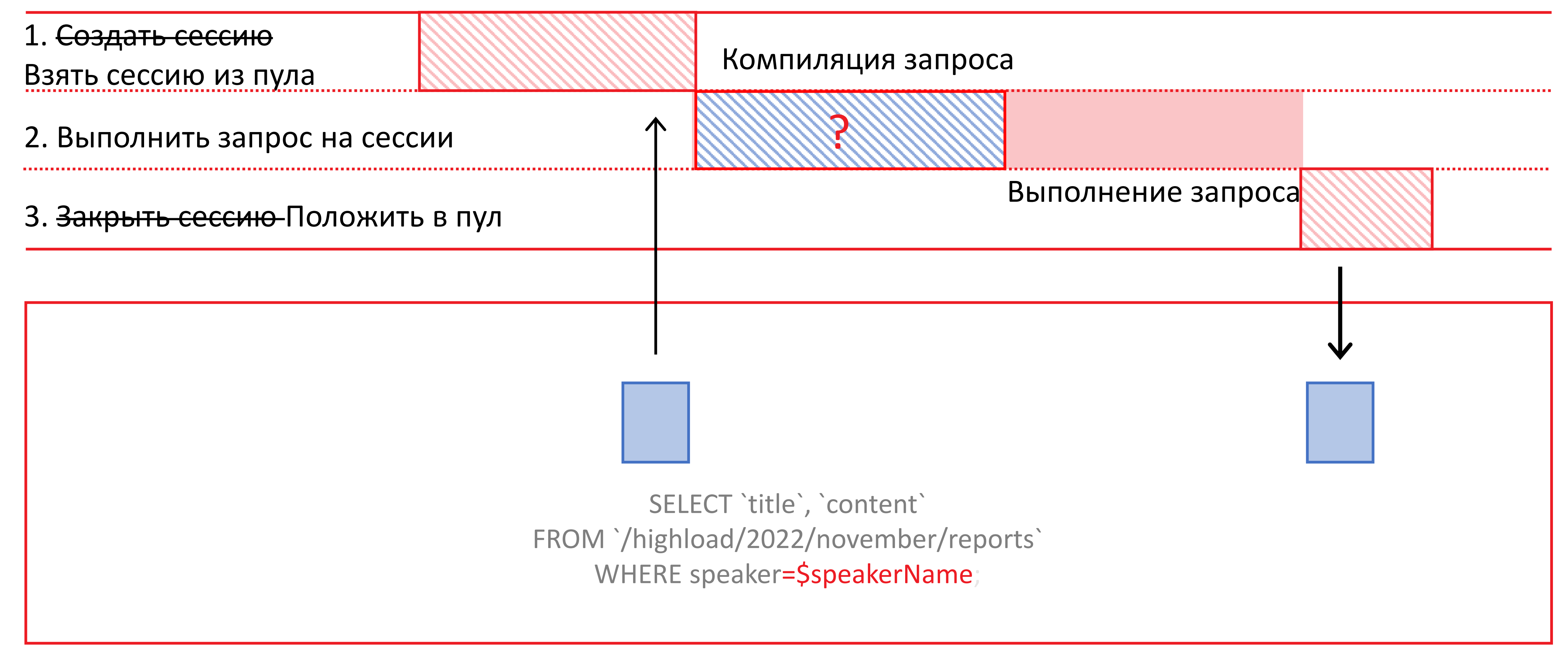
# 3.2 Фоновый *KeepAlive* для простаивающих сессий



# Что еще оптимизировать?

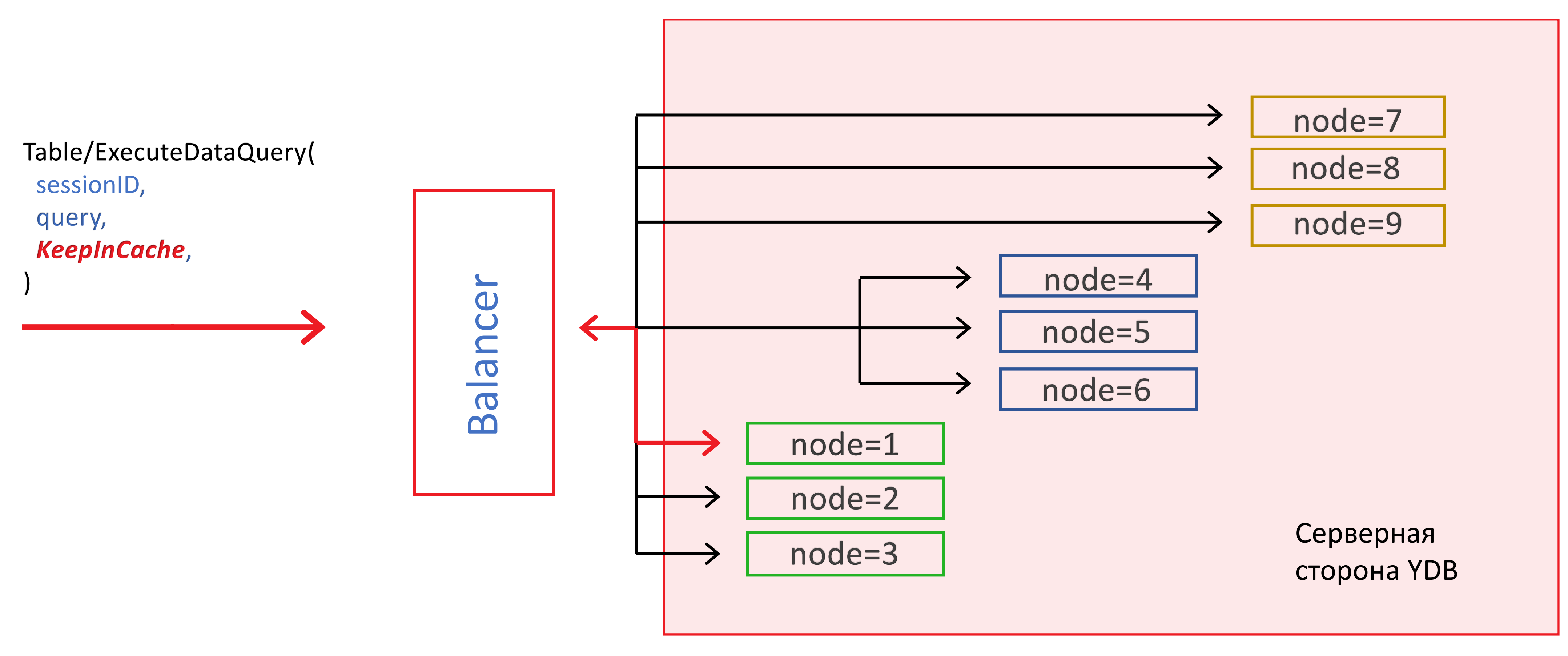


# Что еще оптимизировать?



# 4. Кэширование результатов компиляции запроса

# Флаг кэширования результатов компиляции запроса



# Запрос закэширован на ноде 1

Сессия | Сессия | ... | Сессия
Сессия | Сессия | ... | Сессия
Сессия | Сессия | ... | Сессия
Сессия | Сессия | ... | Сессия
... | ... | | ...
Сессия | Сессия | ... | Сессия

q1

LRU-кэш





# Первый запрос на ноде 1

| | |
|---|---|
| 1. Создать сессию (взять из пула) | |
| 2. Выполнить запрос на сессии | |
| 3. Вернуть сессию в пул | |

# Повторный запрос на ноде 1

1. Создать сессию
(взять из пула)

2. Выполнить запрос на сессии

3. Вернуть сессию в пул

# Флаг кэширования запросов

+ уменьшает общее время выполнения запроса
+ упрощает клиентский код
+ защищает от рестартов нод
+ защищает от вымывания серверного кэша
+ помогает равномерно использовать все ноды базы для запросов
+ включен по дефолту для всех запросов с параметрами

# 5. Серверная балансировка

# 5.1 Сессия может быть создана на другой ноде

Table/CreateSession()

"ydb://session/5678&
**node_id=4**&id=abcdef=="

Balancer

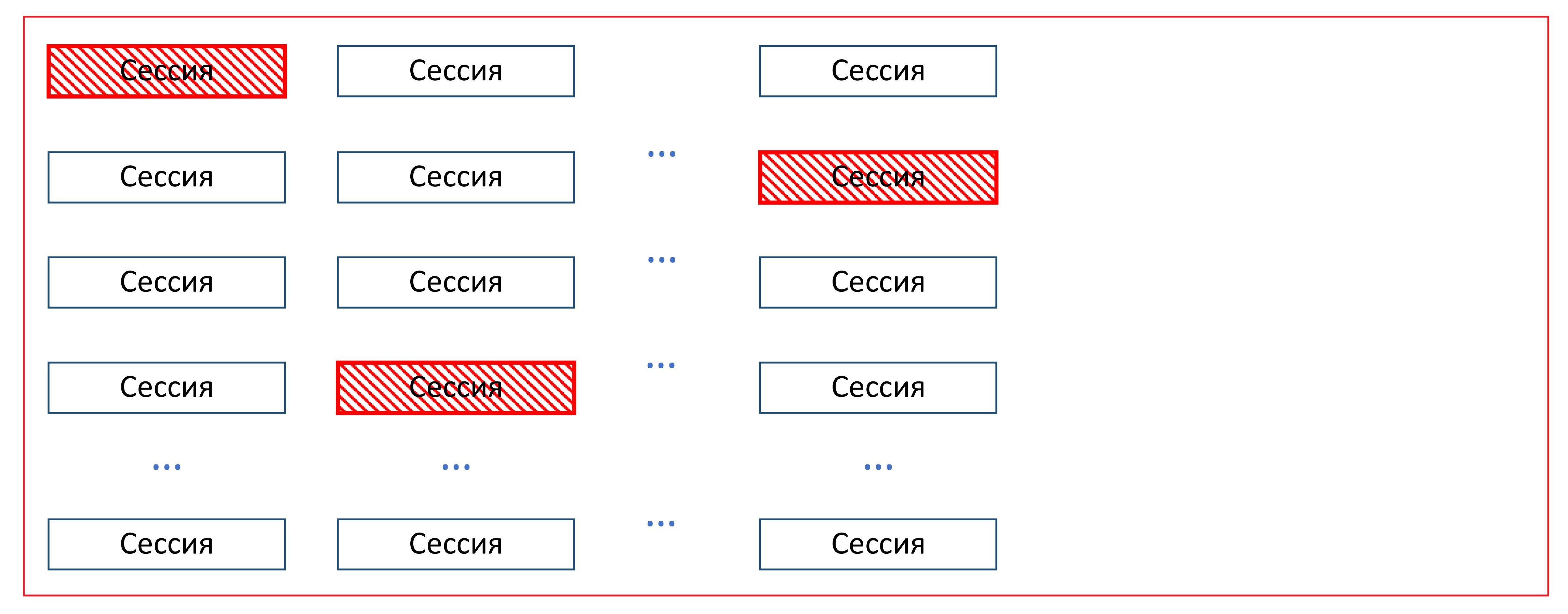
node=7
node=8
node=9
node=4
node=5
node=6
node=1
node=2
node=3

Серверная
сторона YDB





# 5.2 Сессии могут быть закрыты сервером принудительно



# 6. Обработка ошибок

# Ожидаемые ошибки

AZ1

AZ2

AZ3

...

App1 App2 App3 App4 App5 App6 App7 App8





# Наивный ретраер

```
func retry(ctx, db, query) result {
  for {
    s, err := db.Table().CreateSession(ctx)
    if err != nil {
      continue
    }
    result, err := s.Execute(ctx, query)
    if err != nil {
      s.Close()
      continue
    }
    s.Close()
    return result
  }
}
```

# Ретраер с лимитом попыток

```
func retry(ctx, db, query) result {
  for i := 0; i < 10; i++ {
    s, err := db.Table().CreateSession(ctx)
    if err != nil {
      continue
    }
    result, err := s.Execute(ctx, query)
    if err != nil {
      s.Close()
      continue
    }
    s.Close()
    return result, nil
  }
  return nil, fmt.Errorf("no progress")
}
```

# Ретраер с экспоненциальной задержкой

```
func retry(ctx, db, query) result {
  for i := 0; i < 10; i++ {
    if i > 0 {
      time.Sleep(math.Pow(2, i) * time.Millisecond)
    }
    s, err := db.Table().CreateSession(ctx)
    if err != nil {
      continue
    }
    result, err := s.Execute(ctx, query)
    if err != nil {
      s.Close()
      continue
    }
    s.Close()
    return result, nil
  }
  return nil, fmt.Errorf(“no progress”)
}
```

# Транспортные и серверные ошибки

Canceled
Unknown
InvalidArgument
DeadlineExceeded
AlreadyExists
PermissionDenied

Unauthenticated
ResourceExhausted
FailedPrecondition
Aborted
OutOfRange

Internal
Unavailable
DataLoss
Unimplemented

- транспортные ошибки
- серверные ошибки

BAD_REQUEST
UNAUTHORIZED
INTERNAL_ERROR
ABORTED
UNAVAILABLE
OVERLOADED
SCHEME_ERROR

GENERIC_ERROR
TIMEOUT
BAD_SESSION
PRECONDITION_FAILED
ALREADY_EXISTS
NOT_FOUND
SESSION_EXPIRED

CANCELLED
UNDETERMINED
UNSUPPORTED
SESSION_BUSY

Ссылка

# Ретраибельные ошибки

Canceled
Unauthenticated
Internal
Unknown
ResourceExhausted
Unavailable
InvalidArgument
FailedPrecondition
DataLoss
DeadlineExceeded
Aborted
Unimplemented
AlreadyExists
OutOfRange
PermissionDenied

- транспортные ошибки
- серверные ошибки
- можно безопасно ретраить
- можно ретраить, если операция идемпотентная

BAD_REQUEST
GENERIC_ERROR
UNAUTHORIZED
TIMEOUT
INTERNAL_ERROR
BAD_SESSION
ABORTED
PRECONDITION_FAILED
CANCELLED
UNAVAILABLE
ALREADY_EXISTS
UNDETERMINED
OVERLOADED
NOT_FOUND
UNSUPPORTED
SCHEME_ERROR
SESSION_EXPIRED
SESSION_BUSY

Ссылка

# Следует удалить сессию

- транспортные ошибки
- серверные ошибки
- сессия более непригодна (выделено красной рамкой)

Canceled
Unknown
InvalidArgument
DeadlineExceeded
AlreadyExists
PermissionDenied
Unauthenticated
ResourceExhausted
FailedPrecondition
Aborted
OutOfRange
Internal
Unavailable
DataLoss
Unimplemented

BAD_REQUEST
UNAUTHORIZED
INTERNAL_ERROR
ABORTED
UNAVAILABLE
OVERLOADED
SCHEME_ERROR
GENERIC_ERROR
TIMEOUT
BAD_SESSION
PRECONDITION_FAILED
ALREADY_EXISTS
NOT_FOUND
SESSION_EXPIRED
CANCELLED
UNDETERMINED
UNSUPPORTED
SESSION_BUSY

Ссылка

# С «медленной» экспоненциальной задержкой

| | | |
|---|---|---|
| Canceled | Unauthenticated | Internal |
| Unknown | ResourceExhausted | Unavailable |
| InvalidArgument | FailedPrecondition | DataLoss |
| DeadlineExceeded | Aborted | Unimplemented |
| AlreadyExists | OutOfRange | |
| PermissionDenied | | |

- транспортные ошибки
- серверные ошибки
- экспоненциальная задержка между попытками от 1с

| | | |
|---|---|---|
| BAD_REQUEST | GENERIC_ERROR | |
| UNAUTHORIZED | TIMEOUT | |
| INTERNAL_ERROR | BAD_SESSION | |
| ABORTED | PRECONDITION_FAILED | CANCELLED |
| UNAVAILABLE | ALREADY_EXISTS | UNDETERMINED |
| OVERLOADED | NOT_FOUND | UNSUPPORTED |
| SCHEME_ERROR | SESSION_EXPIRED | SESSION_BUSY |

Ссылка

# С «быстрой» экспоненциальной задержкой

Canceled
Unknown
InvalidArgument
DeadlineExceeded
AlreadyExists
PermissionDenied

Unauthenticated
ResourceExhausted
FailedPrecondition
Aborted
OutOfRange

Internal
Unavailable
DataLoss
Unimplemented

- транспортные ошибки
- серверные ошибки
- экспоненциальная задержка между попытками от 5мс

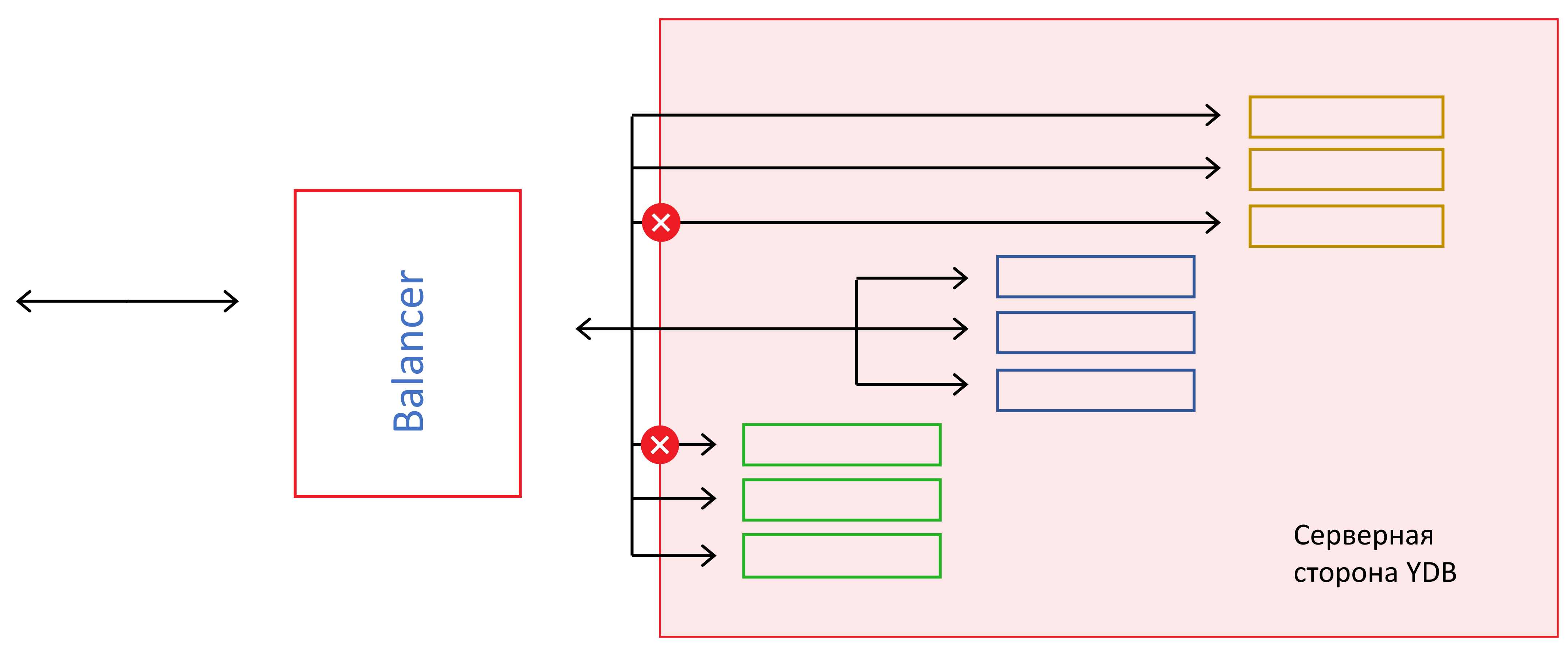
BAD_REQUEST
UNAUTHORIZED
INTERNAL_ERROR
ABORTED
UNAVAILABLE
OVERLOADED
SCHEME_ERROR

GENERIC_ERROR
TIMEOUT
BAD_SESSION
PRECONDITION_FAILED
ALREADY_EXISTS
NOT_FOUND
SESSION_EXPIRED

CANCELLED
UNDETERMINED
UNSUPPORTED
SESSION_BUSY

Ссылка

# Пессимизация соединений



# Пессимизация соединений

Connections usage

80
70
60
50
40
30
20
10
0

19:25:30
19:27:00

**2021-12-29 19:27:00**

| | |
|---|---|
| kikimr0425.search.yandex.net:2135 | 0 |
| kikimr0425.search.yandex.net:31051 | 2 |
| kikimr0447.search.yandex.net:31051 | 2 |
| kikimr0482.search.yandex.net:31051 | 23 |
| kikimr0518.search.yandex.net:31051 | 28 |
| kikimr0602.search.yandex.net:31051 | 25 |





# Пессимизация соединений – путь в один конец?

- Discovery/ListEndpoints === source of truth
- Force re-discovery, если пессимизировано более 50% соединений

# «Умные» ретраеры в драйвере YDB

```
err := db.Table().Do(ctx, func(ctx, session) error {
  result, err := s.Execute(ctx, query)
  if err != nil {
    return err
  }
  var title, content string
  for result.NextResultSet(ctx) {
    for result.NextRow() {
      if err := result.Scan(&title, &content); err != nil {
        return err
      }
      log.Println(title, content)
    }
  }
  return result.Err()
}, table.WithIdempotent(true))
```

# Retry operation возвращает ошибку для обработки на стороне драйвера YDB

```
err := db.Table().Do(ctx, func(ctx, session) error {
  result, err := s.Execute(ctx, query)
  if err != nil {
    return err
  }
  var title, content string
  for result.NextResultSet(ctx) {
    for result.NextRow() {
      if err := result.Scan(&title, &content); err != nil {
        return err
      }
      log.Println(title, content)
    }
  }
  return result.Err()
}, table.WithIdempotent(true))
```

# Сущности, порожденные волатильной сессией, также волатильные

```
var resultOutOfRetryOperation result
err := db.Table().Do(ctx, func(ctx, session) error {
  result, err := session.Execute(ctx, query)
  if err != nil {
    return err
  }
  resultOutOfRetryOperation = result
  var title, content string
  for result.NextResultSet(ctx) {
    for result.NextRow() {
      if err := result.Scan(&title, &content); err != nil {
        return err
      }
      log.Println(title, content)
    }
  }
  return result.Err()
}, table.WithIdempotent(true))
```

# Существенные параметры ретраера

```
err := db.Table().Do(ctx, func(ctx, session) error {
  result, err := s.Execute(ctx, query)
  if err != nil {
    return err
  }
  var title, content string
  for result.NextResultSet(ctx) {
    for result.NextRow() {
      if err := result.Scan(&title, &content); err != nil {
        return err
      }
      log.Println(title, content)
    }
  }
  return result.Err()
}, table.WithIdempotent(true))
```

# Тестирование надежности драйверов YDB

Tasks

| | |
|---|---|
| 1.0 / 0.5 / 0.0 | 22:30 · 23:00 · 23:30 · Nov 21 · 00:30 · 01:00 · 01:30 · 02:00 · 02:30 |







# Драйвер распределенной базы данных YDB

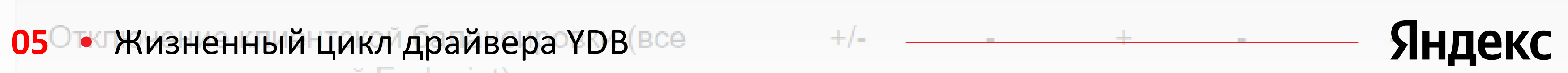
+ умеет инициализироваться
+ запускает фоновый процесс актуализации состояния YDB
+ реализует клиентскую балансировку запросов
+ осуществляет привязку сессий и нод YDB
+ имеет пул сессий и выполняет фоновый KeepAlive
+ корректно обрабатывает ошибки
+ имеет ретраеры
+ поддерживает пессимизацию соединений
+ готов к серверной балансировке

MUST HAVE

# Feature parity

| Feature | C++ | Python | Go | Java | NodeJS | C# | Rust |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Поддержка SSL/TLS (системные сертификаты) | + | + | + | + | + | + | + |
| Поддержка SSL/TLS (кастомные сертификаты) | + | + | + | + | + | | - |
| Возможность настроить/включить GRPC KeepAlive (фоновое поддержание живости соединения) | + | + | + | ? | | | - |
| Регулярный прогон тестов SLO на последней версии кода | + | +/- | + | + | +/- | - | - |
| Шаблоны Issue в GitHub | - | ? | + | - | + | | - |
| **Клиентская балансировка** | | | | | | | |
| Инициализация балансировщика через Discovery/ListEndpoints | + | + | + | + | + | + | + |
| Отключение клиентской балансировки (все запросы в начальный Endpoint) | +/- | - | + | - | | | + |
| Фоновый Discovery/ListEndpoints (раз в минуту | + | + | + | + | + | + | + |

Ссылка на таблицу




# Наши официальные репозитории



https://github.com/ydb-platform/ydb-go-sdk



https://github.com/ydb-platform/ydb-nodejs-sdk



https://github.com/ydb-platform/ydb-rs-sdk



https://github.com/ydb-platform/ydb-python-sdk



https://github.com/ydb-platform/ydb-java-sdk



https://github.com/ydb-platform/ydb-dotnet-sdk



https://github.com/ydb-platform/ydb-php-sdk






**Алексей Мясников**

*старший разработчик в команде YDB*

asmyasnikov@yandex-team.ru

https://t.me/asmyasnikov



https://clck.ru/32kaXr




Яндекс

Голосуйте за доклад

